

Remarks

Applicants respectfully request reconsideration of the application in view of the foregoing amendments and following remarks.

Applicants have canceled claims 2 and 3 without prejudice. Claims 1 and 4-36 are pending. Claim 21 has been rejected under 35 U.S.C. § 102(a) in view of the article Richard Grimes, "Attribute Programming with Visual C++" ["Grimes article"]. Claims 1, 4-20, and 22-36 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over the Grimes article in view of portions of the book Aho et al., Compilers Principles, Techniques, and Tools ["Aho book"]. Applicants respectfully disagree.

I. IDS filed October 8, 2003

Applicants filed an IDS on October 8, 2003, accompanied by a two-page Form 1449. The IDS was received by the USPTO on October 10, 2003. Please provide an initialed Form 1449 for the IDS filed October 8, 2003.

II. Claims 1, 4-20, and 22-36

Claims 1, 4-20, and 22-36 have been rejected under 35 U.S.C. § 103(a) as being unpatentable over the Grimes article in view of the Aho book. The Grimes article and Aho book, taken separately or in combination, fail to teach or suggest at least one limitation from each of claims 1, 4-20, and 22-36.

Claim 1, which has been amended to include language from claims 2 and 3, recites:

the intermediate representation includes a symbol table and a tree that unifies representation of the programming language code and the embedded definition language information, wherein the symbol table includes plural entries for symbol names for the programming language code, and wherein at least one of the plural entries has an associated list of definition language attributes....

Claim 6 recites:

an intermediate representation comprising a symbol table and a parse tree, wherein the symbol table includes plural entries for symbol names for the programming language constructs, at least one of the plural entries having an associated list of interface definition language attributes, and wherein the parse

tree unifies representation of the programming language constructs and the interface definition language constructs.

Claim 9 recites:

a symbol table that has plural entries, each of the plural entries corresponding to a symbol name for programming language code of a file having a combination of programming language code and embedded interface definition language information, at least one of the plural entries having an associated list of interface definition language attributes based upon the embedded interface definition language information; and

...
a parse tree, wherein the parse tree unifies representation of the programming language code and the embedded interface definition language information.

Claim 10, as amended, recites:

an intermediate representation including a tree that unifies representation of the programming language constructs and the definition language constructs....

Claim 13, which has been amended to include language from claim 14, recites:

a tree that unifies representation of the programming language code and the embedded definition language information....

Claim 16 recites:

a symbol table having plural entries for symbol names for the programming language code, at least one of the plural entries having an associated list of definition language attributes based upon the embedded definition language information; and

...
a tree that unifies representation of the embedded definition language information and the programming language code.

Claim 18, as amended, recites:

the intermediate representation includes a tree that unifies representation of the definition language information and the programming language code....

Claim 20, as amended, recites:

the parse tree unifies representation of the interface definition language information and the programming language source code.

The Grimes article and Aho book, taken separately or in combination, fail to teach or suggest the above-cited language from each of claims 1, 6, 9, 10, 13, 16, 18, and 20, respectively.

The Grimes article describes IDL attributes and an [emitidl] attribute that "indicates that IDL should be generated from the attributes and placed into the compiled .obj file." [Grimes article, page 2.] This involves processing the IDL attributes and the generated IDL *apart from C++ code during compilation*, and it involves separating the generated IDL from the C++ code. It does not involve generating computer-executable code (as done from the C++ code), and it does not involve the compiler integrating the IDL and C++ for later compilation. The Grimes article here leads away from a tree that unifies representation of (1) programming language code (or constructs) and (2) definition language information (or constructs) as recited in the above-cited language of claims 1, 6, 9, 10, 13, 16, 18, and 20, respectively. Similarly, the Grimes article here leads away from a symbol table that has (1) entries for symbol names for programming language code (or constructs) and (2) an associated list of definition language attributes for an entry as recited in the above-cited language of claims 1, 6, 9, and 16, respectively.

The Grimes article also describes injecting C++ code in response to attributes generally. [Grimes article, pages 2-3.] According to the Grimes article:

When the compiler sees an attribute in your C++ code it passes the attribute and its arguments to the attribute providers that have been registered on the system. The compiler does this by calling methods on an interface called IAttributeHandler that is implemented by the provider. So that the provider has access to the compiler, it also passes a callback interface pointer of the type ICompiler. If the provider recognizes the attribute, it can then call methods on ICompiler to tell the compiler to add C++ code to replace the attribute.

[Grimes article, page 3; see also the diagram on page 3.] This involves replacing an attribute with C++ code when the compiler encounters the attribute. Replacing an attribute with C++ code when the compiler encounters the attribute does not involve the compiler integrating IDL and C++ for later compilation. The Grimes article again leads away from a tree that unifies representation of (1) programming language code (or constructs) and (2) definition language

KBR:kbr 8/9/04 3382-56062-01 MS 147268.1 284008

PATENT
Atty. Ref. No. 3382-56062-01

information (or constructs) as recited in the above-cited language of claims 1, 6, 9, 10, 13, 16, 18, and 20, respectively. And, the Grimes article again leads away from a symbol table that has (1) entries for symbol names for programming language code (or constructs) and (2) an associated list of definition language attributes for an entry as recited in the above-cited language of claims 1, 6, 9, and 16, respectively.

The Aho book does not teach or suggest the above-cited language from claims 1, 6, 9, 10, 13, 16, 18, and 20, respectively. The Aho book generally describes compiler techniques and tools. The parts of the Aho book specifically cited by the Examiner describe, among other things, compiler organization by "front end" and "back end" [Aho book, page 20], symbol tables [Aho book, page 11], and parse trees [Aho book, pages 6 and 40-48], but do not address working with both programming language and definition language.

The Examiner writes:

It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grimes' system of C++ code and definition code with a compiler, which would generate executable code as found in Aho's teaching. This implementation would have been obvious because one of ordinary skill would be motivated to provide a mechanism, which would allow source code to produce meaningful executable code. Finally, upon the above combination, it can be seen that symbol tables (provided by Aho), would have entries that contain a list of attributes associated with interface definition language (provided by Grimes).

[Final Office action, pages 6-7.] Elsewhere, the Examiner writes:

It would have been obvious to one of ordinary skill in the art at the time of invention to implement Grimes' interface definition language / programming language compiler with symbol tables and parse trees as appropriate for compiling such a combination as found in Aho's teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to use common and well understood techniques for implementing compilers.

[Final Office action, pages 7-8; see also page 14.] Applicants respectfully disagree. Even if, for the sake of argument, the compiler of the Grimes article were to be implemented with a symbol table and parse tree as described in the Aho book (or with any other symbol table and parse tree, as those terms are broadly understood in the art), the symbol table and parse tree would be used for processing C++ code but not for also processing IDL, since the Grimes article describes (1) processing IDL attributes and the generated IDL apart from C++ code during compilation, or (2)

KBR:ktr 8/9/04 3382-56062-01 MS 147268.1 284008

PATENT
Atty. Rcf. No. 3382-56062-01

replacing an attribute, generally, with C++ code when the attribute is encountered by the compiler.

Claims 1, 6, 9, 10, 13, 16, 18, and 20 should be allowable. In view of the foregoing discussion, Applicants will not belabor the merits of the separate patentability of dependent claims 4, 5, 7, 8, 11, 12, 14, 15, 17, 19, and 22-36.

III. Claim 21

Claim 21 recites:

embedding by the programming language compiler debugging information in a definition language output file, the definition language output file for subsequent processing by a definition language compiler....

The Grimes article fails to teach or suggest the above-cited language from claim 21. The Examiner writes: "The broadest reasonable interpretation of claim 21 does not force a particular definition of 'a definition language output file'. The project's PDB file is read on by claim 21 and so is the attributes being placed in the .obj file." Applicants disagree with this characterization of the Grimes article. In any case, even if the PDB file of the Grimes article includes debugging information, the PDB file is not a "file for subsequent processing by a definition language compiler," as recited in claim 21. With its description of a PDB file, the Grimes article does not teach or suggest the above-cited language of claim 21. As for the .obj file of the Grimes article, it does not include debugging information, and the Grimes article again does not teach or suggest the above-cited language of claim 21.

Claim 21 should be allowable.

Conclusion

Claims 1 and 4-36 should be allowable. Such action is respectfully requested.

Request for Interview

In view of the preceding amendments and remarks, Applicant believes the application to be allowable. If any issues remain, however, the Examiner is formally requested to contact the undersigned attorney at (503) 226-7391 prior to issuance of the next communication in order to

KBR:kbr 8/9/04 3382-56062-01 MS 147268.1 284008

PATENT
Atty. Ref. No. 3382-56062-01

arrange a telephonic interview. This request is being submitted under MPEP § 713.01, which indicates that an interview may be arranged in advance by a written request.

Respectfully submitted,

KLARQUIST SPARKMAN, LLP

By



Kyle B. Kinehart
Registration No. 47,027

One World Trade Center, Suite 1600
121 S.W. Salmon Street
Portland, Oregon 97204
Telephone: (503) 226-7391
Facsimile: (503) 228-9446
(147268.1)